ORIGINAL RESEARCH

Real-time detection of moving cast shadows using foreground luminance statistics

Mehran Andalibi*[1], Lawrence Hoberock[2], Damon M. Chandler[3]

[1] *Department of Aerospace and Mechanical Engineering, Embry-Riddle Aeronautical University, Prescott, AZ, U.S.A*
[2] *School of Mechanical and Aerospace Engineering, Oklahoma State University, Stillwater, OK, U.S.A*
[3] *Department of Electrical and Electronic Engineering, Shizuoka University, Hamamatsu City, Shizuoka, Japan*

Received: November 11, 2015 **Accepted:** January 3, 2016 **Online Published:** January 20, 2016
DOI: 10.5430/air.v5n1p144 **URL:** http://dx.doi.org/10.5430/air.v5n1p144

ABSTRACT

This paper presents a robust real-time method for detection of moving cast shadows which employs the assumption of higher interdependence of luminance values for the shadow pixels in larger regions compared to the object pixels. First, a fast modified image differencing technique is used to separate foreground pixels from the background. Next, for a moving window of fixed width scanning the foreground regions, a new measure called Modified Correlation is introduced. The new measure is determined by first computing the correlation between the luminance values of the moving window and luminance values of its neighbouring windows; this correlation is then divided by a robust-to-noise range measured based on the luminance values of the moving window. The modified correlation exhibits abrupt rising transitions as it enters the shadow region from the object region, transitions which can be used to separate object pixels from shadow pixels. Thus, the new method is very effective at suppressing moving cast shadows, while avoiding limiting structures, unrealistic assumptions, the need for a-priori knowledge, and manual selection of critical parameters. An average shadow detection rate of 85.4% and an average shadow discrimination rate of 99.5% over multiple different sequences, higher than those of available methods in the literature, confirm the efficacy of the method. The robustness of the method is examined under different lighting conditions, different target-environment combinations, and sequences with object-shadow occlusion. The proposed method is computationally efficient and suitable for real-time situations.

Key Words: Moving cast shadow, Image difference, Foreground mask, Sliding window, Correlation, Real-time detection

1. INTRODUCTION

1.1 Problem introduction and importance

It is a routine task for the human eyes to immediately distinguish shadows from objects, but identifying shadows by a computer algorithm is a challenging problem.[1] Shadow pixels and object pixels share two important visual features: 1) motion dynamics and 2) detectability.[2] For instance, to detect a moving object, a surveillance system might estimate the foreground via image differencing. However, the shadow cast by the moving object will also be detected, since it is moving. The cast shadow makes it difficult to detect the exact shape of the object. Segmentation techniques would also fail to localize cast shadows, because cast shadows are adjacent to object points.[3] Shadows cause serious problems while segmenting and extracting objects, due to the misclassification of shadow points as foreground. Shadows can cause object merging, object shape distortion and even object

*Correspondence: Mehran Andalibi; Email: andalibm@erau.edu; Address: Department of Aerospace and Mechanical Engineering, Embry-Riddle Aeronautical University, 3700 Willow Creek Rd, Prescott, AZ, U.S.A.

losses.[4,5] Therefore, accurate detection of a moving object and acquisition of its exact shape by eliminating shadows has a great effect on the performance of subsequent steps, such as tracking, recognition, classification, and activity analysis.[6]

1.2 Relevant literature

Prati *et al.*[7] classified shadow detection algorithms by using a two-layer taxonomy, namely deterministic and statistical approaches, based on whether the decision process introduces and exploits uncertainty. The taxonomy further classifies deterministic approaches into model-based and nonmodel-based methods, and probabilistic approaches into parametric and nonparametric methods. An alternative classification of cast-shadow detection algorithms is proposed in Ref.,[8] based on object/environment dependency and the implementation domain of the algorithm (pixel or transform domain).

Although a large volume of literature exists in the field of moving cast-shadow suppression, many of which are sophisticated and accurate,[3,9] there are limitations common to most. A general limitation is the real-time implementation of the method, since the more sophisticated and accurate is the algorithm, the larger is the required computation time, which impairs application of the method for many real-time situations. In some statistical methods, manual parameter selection is a critical issue,[9–11] which prevents unsupervised implementations. The need for a-priori knowledge, such as the height of pedestrians[12] (which is not available in many cases), manual segmentation, and non-rigorous assumptions, such as the linear transformation of pixel RGB values after being covered by a shadow,[12] are other disadvantages, even though they may provide accurate results.

In deterministic methods, a model-based approach might achieve the best results, but it is often too complex[13,14] and time-consuming compared to nonmodel-based methods. In this category, the number and complexity of the models increase rapidly if the aim is to deal with complex and cluttered environments with different lighting conditions, object classes, and perspective views. Also, many of the model assumptions used in this category cannot be confirmed in real-world conditions, such as the assumption of a Gaussian distribution for the light intensity of background pixels in Ref.[15]

Some deterministic nonmodel-based methods might be implemented in real-time situations, and do not depend on a-priori knowledge. However, some methods are suited only for specific environments (indoor/outdoor) and specific targets (human/vehicle/others).[15–17] For instance, Ref.[17] only deals with detection of vehicle cast shadows. Some publications employ assumptions that do not match real-world conditions, such as assuming the presence of a strong light source,[2,18] and neglecting the penumbra.[2] Furthermore, the features used for shadow detection are not always robust and reliable, such as small changes in hue.[2] Stauder *et al.*[18] suggest a shadow model used in many publications, including Refs.[2,3,19] This method provides a basis for shadow detection and is one of the earliest publications that considers and models light intensity variation within the shadow region and the penumbra.

It worth mentioning that while researchers typically utilize colour information as well as light intensity, texture, or other cues to detect cast shadows, the authors have examined multiple methods which assumed chromaticity invariance of a region when covered by a cast shadow.[3,20–22] It was concluded that these assumptions are not always valid and robust for different types of applications and illumination conditions. As observed by the authors, in most of the compressed videos,[3] when the light source is not white or there is colour blending among objects in the scene, the chromaticity invariance assumption cannot always be trusted. Therefore in our proposed algorithm, we only utilize luminance values. This also helps maintain the computation time smaller.

1.3 Proposed approach

In our proposed method, called the AHC method, which is classified as a deterministic nonmodel-based approach, we overcome some of the abovementioned disadvantages, namely the considerable computation time, the need for a-priori knowledge, the manual selection of parameters, and limiting structures and assumptions. AHC employs the assumption of higher interdependence of luminance values for shadow pixels in larger regions and consists of two consecutive stages. During the first stage, we detect foreground pixels using a fast modified image differencing technique. In the second stage, we introduce a new measure called “Modified Correlation” which is calculated for each moving window scanning the foreground regions; the modified correlation is computed by dividing the correlation between the luminance values of the moving window and its neighbouring windows by a robust-to-noise range of luminance values for the moving window. Modified correlation exhibited a significant drop as the moving window enters the object region from the shadow region, while maintaining small variations within each region, so it is utilized to segment the foreground pixels into shadow and object regions.

The main contributions of this work are: 1) Remarkably small computation time and implementation of the algorithm in MATLAB and also in C++ using OpenCV libraries for real-time videos; 2) Avoiding unrealistic models and assumptions, which increases the robustness of the method under different conditions; 3) Avoidance of a-priori knowledge, manual selection of crucial parameters, and application-dependence, which makes an unsupervised implementation feasible; 4) Higher detection accuracies compared to existing approaches for benchmark sequences.

Our proposed AHC algorithm is described in Section 2. Qualitative results for recorded sequences are provided in Section 3.1. Evaluations and comparisons for benchmark videos and discussion of results are presented in Section 3.2 and 3.3, respectively. A performance analysis of a real-time implementation of AHC is provided in Section 4. Limitations and future work are proposed in Section 5, and conclusions are included in Section 6.

2. Methods

The algorithm used to detect moving cast shadows includes two consecutive stages: 1) Foreground detection in which a mask for moving pixels is created; 2) Suppression of the cast shadow using modified correlation. Figure 1 gives the flowchart of the proposed algorithm.

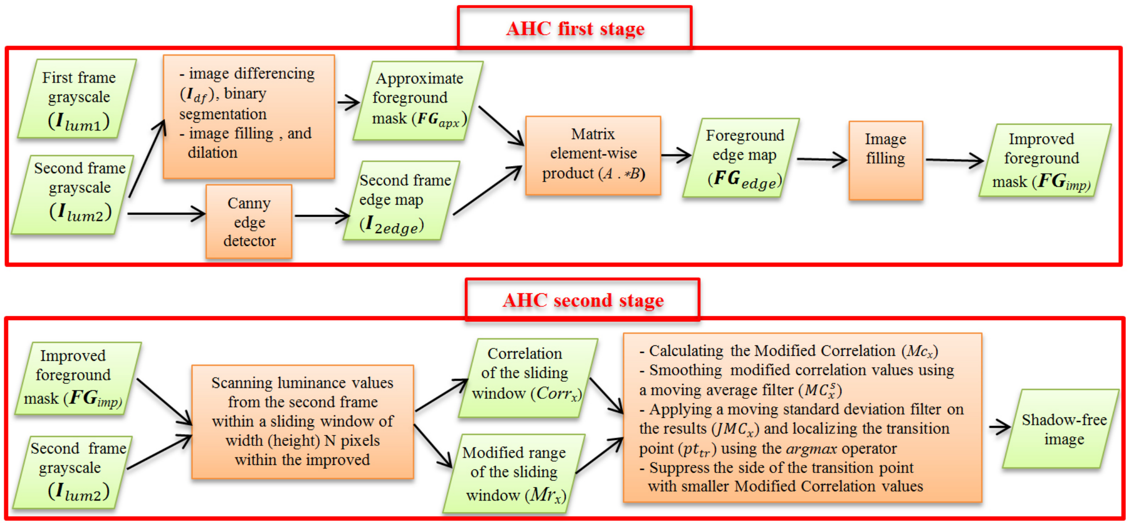

AHC first stage
First frame grayscale (I_{lum1})
Second frame grayscale (I_{lum2})
- image differencing (I_{df}), binary segmentation
- image filling , and dilation
Approximate foreground mask (FG_{apx})
Canny edge detector
Second frame edge map (I_{2edge})
Matrix element-wise product (A .* B)
Foreground edge map (FG_{edge})
Image filling
Improved foreground mask (FG_{imp})
AHC second stage
Improved foreground mask (FG_{imp})
Second frame grayscale (I_{lum2})
Scanning luminance values from the second frame within a sliding window of width (height) N pixels within the improved
Correlation of the sliding window ($Corr_x$)
Modified range of the sliding window (Mr_x)
- Calculating the Modified Correlation (Mc_x)
- Smoothing modified correlation values using a moving average filter (MC_x^s)
- Applying a moving standard deviation filter on the results (JMC_x) and localizing the transition point (pt_{tr}) using the argmax operator
- Suppress the side of the transition point with smaller Modified Correlation values
Shadow-free image

Figure 1. Flowchart for the AHC method

In stage 1, after converting the first and second frames to grayscale via a standard grayscale conversion method, image differencing (subtracting consecutive frames) is used to acquire an approximate foreground mask. Binary segmentation might be needed if more than one moving blob exists in the frames. After binary filling and dilation of the approximate foreground mask, it is combined with an edge map of the second frame (acquired using the Canny edge detector) to yield an edge map of the foreground, which only contains the edge pixels of moving objects and cast shadows. Here, an element-wise product (elements are the same as pixels for each image) is used for combining the second frame edge map and the approximate foreground mask. Image filling is then employed on the resulting foreground edge mask to obtain the improved foreground mask that includes only shadow and object interior and boundary pixels.

In stage 2, a sliding window with a width (and/or a height) of a fixed number of pixels scans the luminance values of the improved foreground mask pixels from stage 1 as it moves horizontally (and/or vertically) within the boundaries of this mask. Scanning here refers to reading the luminance values for the pixels within the window from the second frame grayscale and storing them in a vector. At each location of the window, the Pearson correlation coefficient between the window's luminance vector and the luminance vector of a neighbouring window is calculated. This correlation coefficient shows higher values in the shadow region. As another measure to distinguish between the cast shadow and moving objects, a robust-to-noise range measure is calculated via the difference between upper 97 percentiles and the lower 3 percentiles of the sorted luminance values in the stored vector, which shows lower values in the shadow region. To combine and reinforce the effects of both measures and make it easier to distinguish a shadow region from a moving object region, correlation values are then divided by the range values to give the "Modified Correlation" for each window. As the moving window sweeps the improved foreground mask, these modified correlation values calculated for each location of the window are stored in a vector. This modified correlation vector exhibits an abrupt rising transition as the sliding window enters from the object region into the shadow region, while it maintains small variations within each region.

To localize the transition point between the shadow and object regions (abrupt transition), first, the modified correlation values are smoothed by a moving average filter. Then, a moving standard deviation filter is used to the smoothed modified correlation values and the argmax operator applied to the resulting vector provides the location of the transition point. Because shadow regions exhibit higher values of modified correlation, the side of the transition point with a lower average modified correlation corresponds to the object and can thus be suppressed. More details on each step are explained in the following subsections.

2.1 Creating the foreground mask

Any sequence with moving shadows includes three types of pixels: 1) Background pixels; 2) Object pixels (and self-shadow pixels); 3) Cast shadow pixels. However, only object pixels are to be kept for accurate tracking. Many algorithms separate stagnant background pixels from foreground moving pixels (Types 2 and 3) using foreground detection methods, including optical flow or image differencing.[20] Optical flow was not used in this work, because of the considerable computation time required, which prevents real-time applications.

We adopted image differencing with modifications to remove the background pixels in a sufficiently small running time according to the following algorithm, which is illustrated by applying it to two consecutive frames of the Pedestrian sequence (images (a) and (b) of Figure 2 with the first frame on the left):

(1) Convert two consecutive frames (I_1 and I_2) into luminance images (I_{lum1} and I_{lum2}) according to (1), where R, G, and B are RGB channel values, and the weights employed are explained in Ref.[23]

$$I_{lum} = 0.299R + 0.587G + 0.114B \tag{1}$$

It should be noted that grayscale images can be acquired using different colour conversions and from different colour spaces. The simplest and fastest conversion is to take the average of R, G, and B values. Equation (1) is a fast conversion similar to this approach, while it also considers the differences in perception of red, green, and blue colours by human eyes.[23,24] Light intensity values from HSV, HIS, YC_bC_r, YUV, CIEXYZ, and CIELab have also been examined by the authors in order to determine if they can provide any improvement in shadow detection and discrimination rates. The difference in performance was negligible, while these colour conversions added to the computation time, which is of vital importance to the proposed real-time algorithm. Furthermore, for performing some of the conversions accurately, information about the capture device is necessary, which cannot be obtained for the recorded benchmark videos used in this research.

(2) Perform subtraction on the resulting luminance images to acquire the image difference I_{df} given by (2):

$$I_{df} = I_{lum2} - I_{lum1} \tag{2}$$

(3) Apply a threshold to create an approximate binary foreground mask (FG_{apx}) according to (3). Use a small threshold to avoid losing significant number of foreground pixels:

$$\boldsymbol{FG}_{apx} = \begin{cases} 1 & if \quad \boldsymbol{I}_{df} \geq 0.1 \max(\boldsymbol{I}_{df}) \\ 0 & otherwise \end{cases} \tag{3}$$

Then, perform image filling to compensate for foreground pixels left undetected due to overlapping, followed by image dilation. Image dilation is used to avoid losing edge pixels of the second frame, which will be utilized in the next step. The result is displayed in image (c) of Figure 2.

(4) Since the binary foreground mask in the previous step is larger than the real object size (due to object motion and using image differencing), some background pixels will be left inside this mask, which causes misclassifications in future steps. To further remove background pixels, extract the edge map of the second frame (I_{2edge}) using "Canny" edge detector (see Figure 2d), and perform a pixel-wise multiplication to acquire the foreground edge map (FG_{edge}) according to (4) (see Figure 2e):

$$\boldsymbol{FG}_{edge} = \boldsymbol{FG}_{apx} \ .* \ \boldsymbol{I}_{2edge} \tag{4}$$

where $A * B$ means pixel-wise multiplication of matrices A and B. Filling inside this edge map will make an improved binary mask (FG_{imp}), displayed in image (f) of Figure 2.

It is worth mentioning that many algorithms,[3] assume the existence of a fixed background image which is not always available, so they have to update this image which adds to the computation time and fails to provide an accurate updated background. Therefore, we only used consecutive frames.

Comparing the improved foreground map in image (f) with the second frame in image (b) reveals that the resulting map is more accurate than the approximate foreground map in image (c). Although because of imperfections in edge detection and the subsequent image filling procedure, the final map in (f) still contains a limited number of erroneous pixels;

nonetheless, the accuracy in shadow detection remains fairly high, as we will demonstrate in Section 3.

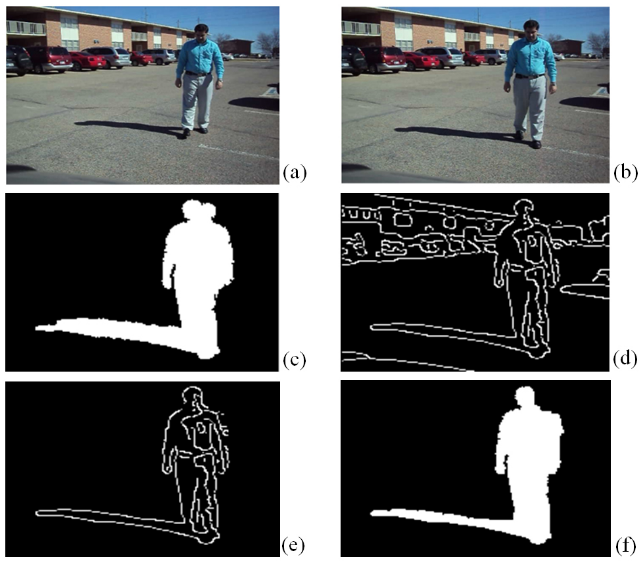


Figure 2. Flowchart for the AHC method

2.2 Suppression of the moving cast shadow

After separating the foreground pixels from the background, the next stage is to distinguish between the shadow and the object. To be able to explain this stage in our algorithm in more detail, we first provide an overview of a few related contributions.

Shadows are created whenever an object blocks light rays that would otherwise strike a surface. Shadow regions have a darker part, which receives no light from the light source (umbra), and a part surrounding the umbra that receives a portion of light from the source (penumbra), depending on the scene and object geometry. Shadows do not usually change the hue or reflectance of the area they physically cover (if they are achromatic), but alter only the irradiance (E), which might be modelled according to (5) proposed by Stauder *et al.*:[18]

$$E(x,y) = \begin{cases} C_A + C_P \cos\left(\angle(N(x,y),L)\right) & if\ \ illuminated, \\ C_A + C_P k(x,y) \cos\left(\angle(N(x,y),L)\right) & if\ \ penumbra, \\ C_A & if\ umbra \end{cases} \tag{5}$$

where C_A and C_P are the intensities of ambient and light sources, respectively; L represents the direction of the light source; and $N(x,y)$ represents the normal direction to the surface at position (x,y). The term $0 \leq k(x,y) < 1$ accounts for the lighting transition inside the penumbra. This transition factor depends on the geometries of light source and scene, and is characterized by slow spatial variation.[25]

There are five assumptions[18] concerning (5): 1) The light source is strong; 2) Camera and background are both static; 3) Background is a textured plane; 4) The light source position is distant from the background; 5) Cast shadows have penumbras. Although assumptions (1), (2), and (4) are often unrealistic, assumptions (3) and (5) hold for many moving cast shadow sequences. Assuming Lambertian reflectance,[26] the luminance $L(x,y)$ of a pixel at position (x,y) can be calculated according to (6):[18]

$$L(x,y) = E(x,y)\rho(x,y) \tag{6}$$

where $\rho(x,y)$ is the reflectance for the pixel at position (x,y). In many real-life sequences, shadows are cast on nearly planar surfaces, which results in small variations of the vector $N(x,y)$ normal to the background pixels. Furthermore, two spatially-neighbouring pixels at positions (x,y) and $(x+\Delta x, y+\Delta y)$ would be expected to have values of $k(x,y)$ approximately the same.[25] As a result, since a shadow does not change the Reflectance, luminance would be constant in a small region within a cast shadow,[3] such that from (5) and (6), we obtain:

$$\begin{aligned} &\rho(x,y) \approx \rho(x+\Delta x, y+\Delta y) \\ &\mathrm{k}(x,y) \approx k(x+\Delta x, y+\Delta y) \\ &N(x,y) \approx N(x+\Delta x, y+\Delta y) \\ &=> L(x,y) \approx L(x+\Delta x, y+\Delta y) \end{aligned} \tag{7}$$

Object surfaces, on the other hand, are usually composed of smaller regions with different reflecting angles, and it leads to a more complicated pattern for irradiance distribution within the object.

The work by Amato *et al.*[3] employs the distinguishing property of cast shadows mentioned earlier. However, since the constancy assumption in (7) is valid only for a small neighbourhood, Amato *et al.* applied gradient-space-connected neighbourhoods (GSCN) segmentation to combine multiple small neighbourhoods detected as belonging to either a shadow or an object. Disadvantageous of this approach are: 1) Assumption of colour constancy for a shadow region instead of luminance constancy; 2) Need to have a fix background; 3) Calculating multiple thresholds from data (higher computation time); 4) Intrinsic imperfection of segmentation for attaching multiple regions.

In AHC, we attempted to detect connected areas of shadow and object instead of detecting small disjoint blobs of shadow and object and then using time-consuming imperfect segmentation methods to connect them. We resorted to statistical measures calculated for larger regions within the boundaries of the foreground to avoid using segmentation, and to avoid failure of (6) for non-small values of Δx and Δy. We observed that although inside a larger shadow region, luminance values of pixels cannot be considered constant, the luminance values exhibit spatial correlations, such that knowing one pixel's luminance can help predict the luminance of the other pixels in a relatively large neighbourhood, to a good approximation. This observation can be deduced from the procedure that leads to (6). In a larger region, $N(x, y)$ and $\rho(x, y)$ exhibit small variations, if the support surface for the shadow is planar. Furthermore, since $k(x, y)$ shows smooth spatial variation, $L(x, y)$ is expected to change smoothly over a shadow region, and have higher self-correlation for a shadow region. Therefore, as one of the statistical measures, we investigated correlation, which is a quantitative indication of how correlated is a pixel's luminance to its neighbouring luminance values.

Here, we use a vertical sliding window (moving from left to right) with a fixed width of N pixels and a varying height that is constrained to the upper and lower boundaries of the improved foreground mask. Images (a) and (b) of Figure 3 illustrate the sliding window (using $N = 8$) within the shadow and object regions of the Pedestrian sequence in Figure 2, respectively. As can be seen, the size (height) of the window changes according to the local foreground edge map variations.

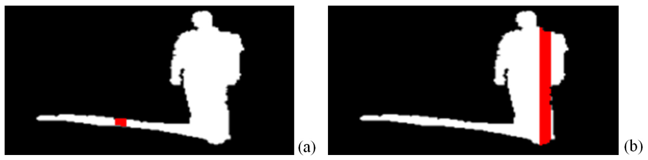


Figure 3. Vertical sliding window within the boundaries of the improved foreground mask. a) An adaptable-height sliding window inside the shadow in the Pedestrian sequence; b) An adaptable-height sliding window inside the object in the Pedestrian sequence

Next, we calculate the Pearson Correlation Coefficient between the luminance values of the sliding window and a neighbouring window. Let L_x denote the vector consisting of luminance values for the sliding window at location x, and let $L_{x,d}$ denote a similar vector for the window shifted d pixels to the right. In order to calculate ρ between L_x and $L_{x,d}$, we need to have vectors of identical size. Therefore, we first trim these vectors using the minimum size of both denoted by sz. Denoting the trimmed vectors by L_x^t and $L_{x,d}^t$, the correlation for the sliding window at location x, $Corr_x$, is computed as:

$$Corr_x = \frac{\sum_{i=1}^{sz}(L_x^t - \overline{L_x^t})(L_{x,d}^t - \overline{L_{x,d}^t})}{\sqrt{\sum_{i=1}^{sz}(L_x^t - \overline{L_x^t})^2}\sqrt{\sum_{i=1}^{sz}(L_{x,d}^t - \overline{L_{x,d}^t})^2}} \tag{8}$$

A discussion of the selection of N and d is provided in Section 3.3.

Part (a) of Figure 4 shows $Corr_x$ plotted versus the centre position of the sliding window, x, as it sweeps the foreground edge map of the Pedestrian sequence. Here $d = 1$ pixel is used for the window shift. As can be observed, there is a decreasing transition in the correlation measure as the sliding window moves from the shadow region to the object region shown by the vertical red line.

In addition to using correlation as a distinguishing factor between a shadow region and an object region, we have considered the range of luminance values. According to Ref.[3] and other researches, a cast shadow reduces the dispersion of luminance values for the background pixels on which it is cast. It has been shown that range and standard deviation are considerably sensitive to outliers in the sample. Therefore, researchers typically utilize other methods of measuring dispersion in a data set, such as Interquartile Range (IQR) or Median Absolute Deviation (MAD). Here, we opted a "Modified Range" similar to IQR which considers more data points in a sample than the middle 50%. We defined the modified range to be the difference between the data point at the $100 - \Delta$ and the Δ percentiles. Denoting L_x sorted ascendingly by L_x^s, the modified robust-to-noise range for the sliding window at location x, MR_x, can be calculated by:

$$MR_x = L_x^s(100 - \delta) - L_x^s(\delta) \tag{9}$$

To avoid discarding significant amount of data points, we use a value of $\Delta = 3$ here.

Part (b) of Figure 4 shows MR_x plotted versus x. As can be observed, there is a relatively abrupt decreasing transition in the range measure as the sliding window moves from the shadow region to the object region shown by the vertical red line.

Since both statistical measures introduced so far exhibit transitions where the sliding window enters the shadow region from the object region or vice versa, we considered combining these measures into a single measure which we call "Modified Correlation" to further reinforce the abrupt transitions seen in both measures. Since a shadow region exhibits a high correlation with its neighbouring windows and exhibits a a small modified range, we define the modified correlation,

MC_x, for the window at location x as follows:

$$MC_x = \frac{Corr_x}{MR_x} \tag{10}$$

As can be seen in part (c) of Figure 4, the modified correlation shows a more abrupt transition than the two contributing measures, and the transition point in the plot approximately matches the real transition point.

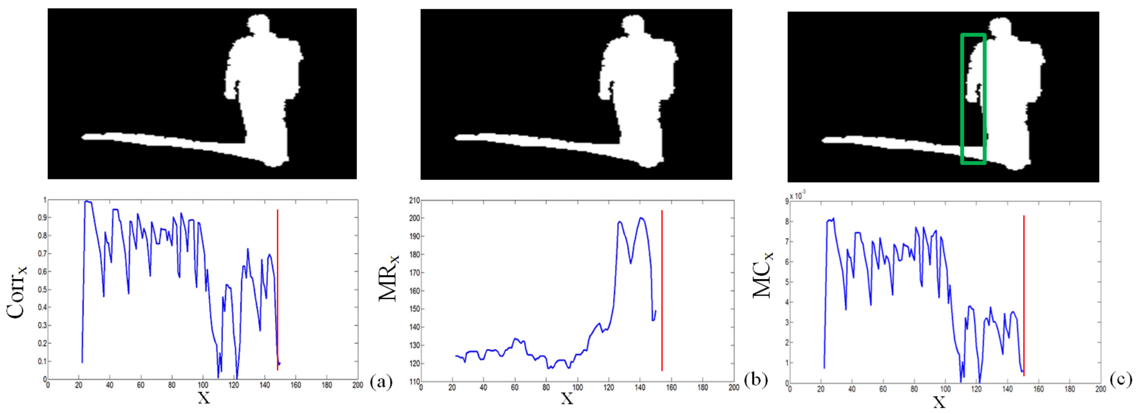


Figure 4. Luminance statistics of the vertical sliding window of width $N = 8$ pixels for the Pedestrian sequence. a) Correlation of the luminance values of a sliding window at location x with a neighbouring window shifted one pixel to the right versus the location of the window centre; b) Modified range for a sliding window at location x versus the location of the window centre using $\delta = 3$; c) Modified correlation for a sliding window at location x versus the location of the window centre

The reason for the approximate match is that when the sliding window falls within the green ractangle shown in the top image of Figure (4c), it will contain shadow, object, and background pixels. Hence, the correlation among adjacent windows is reduced and the range of luminance values within the sliding window increases. This misidentification can happen for objects with concave boundaries due to using merely vertical (and/or horizontal as shown later) sliding windows in AHC; this issue is further discussed in Section 3.3.

Although MC_x shows smaller variations within the shadow or object region compared to the abrupt change at the transition point, we can post-process MC_x values to smoothen the signal to better render the transition point. To do so, we apply a moving average filter of length M to the MC_x values to get the smooth modified correlation for each window denoted by MC_x^s:

$$MC_x^s = \frac{\sum_{i=x-\frac{M}{2}}^{i=x+\frac{M}{2}} MC_i}{M} \tag{11}$$

The length of the moving average filter was empirically chosen to be 10% of the length of modified correlation vector. The result of smoothening the modified correlation values can be seen in part (b) of Figure 5 for the Pedestrian sequence, while the second frame of this sequence is shown in part (a). A clear transition is observable in the MC_x^s plot (here, the first five and last five points are discarded to compensate for the sudden changes at the beginning and the end of signal).

To determine the location of the transition point, denoted by pttr from the MC_x^s plot while avoiding local minima, we applied a moving standard deviation filter of length Q to the MC_x^s values, which highlights the locations at which abrupt variation in the values occur. The results of this filtering, denoted by JMC_x for each window, is given by:

$$JMC_x = \frac{\sqrt{\sum_{i=x-\frac{Q}{2}}^{i=x+\frac{Q}{2}} (MC_i^s - \overline{MC_x^s})^2}}{Q} \tag{12}$$

where $\overline{MC_X^s}$ is average of the vector formed by all MC_i^s values for the window locations at $i \in [x - \frac{Q}{2}, x + \frac{Q}{2}]$.

Q can be determined in a fashion similar to M. The result of applying this filter can be seen in part (c) of Figure 5, where the transition point can be clearly observed and it coincides with the location of the abrupt change in the smooth modified correlation values.

Finally, location of the transition point, pt_{tr}, can be found by (13):

$$pt_{tr} = arg\max_i \; JMC(i) \tag{13}$$

Similar abrupt transitions were observed in other sequences used in this research (shown in the Results Section), as expected. After localization of the transition point using (13), the final segmentation of foreground pixels is performed, where the pixels on the side of transition point with larger average of MC_x^s values are labelled as shadow pixels (depicted green) and the pixels on the other side of transition points with smaller average of MC_x^s values are labelled as object pixels (depicted red) in part (d) of Figure 5.

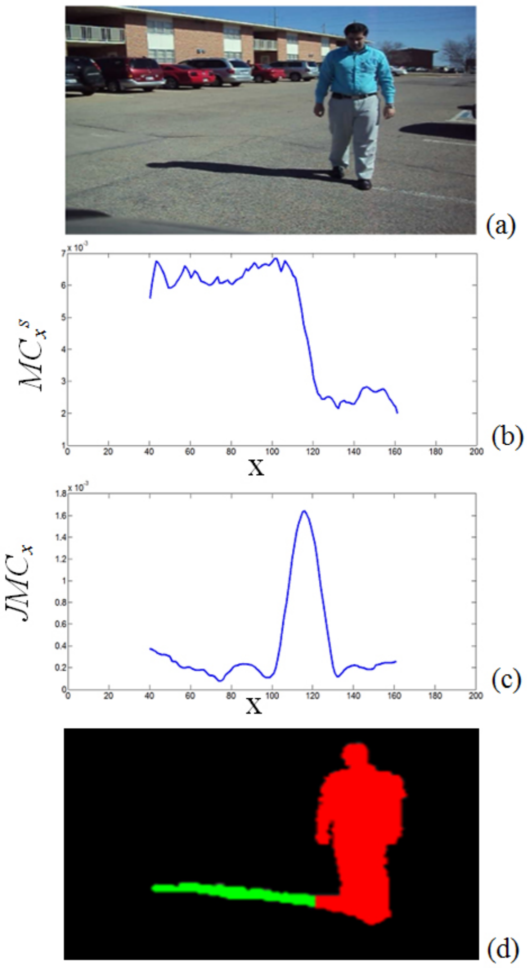


Figure 5. Using modified correlation to segment the foreground. a) Second frame of the Pedestrian Sequence; b) Smooth modified correlation using an Average filter of length 14 *vs.* centre position of the sliding window; c) Localizing the transition point via applying a moving standard deviation filter MC_x^s values; d) Final segmentation of the foreground (Moving cast shadow is depicted in green and the object in red)

3. Results and discussion

The performance and robustness of the method was examined under different combinations of illumination strength, target, and environment for recorded videos, discussed in Section 3.1. Evaluation of the performance of the new method and comparison with available methods in the literature is presented in Section 3.2. A discussion of the results is provided in Section 3.3.

3.1 Qualitative results

We applied the proposed method to a number of different appropriate cases, including indoor and outdoor scenes with human and non-human objects, and different illumination/contrast conditions to examine robustness and efficiency, and to investigate object-environment independence. Throughout this paper, HAC will be tested on 8 different sequences, two of which are benchmark sequences used in many publications for comparison,[3,7,8,27] as well as in Section 3.2 in this research.

Six of the sequences mentioned above can be seen in Figure 6. Note that these frame pairs have different resolutions, but they are displayed using similar sizes here. The first sequence in Figure 6 is the Highway sequence, an outdoor scene with a non-human target, where the shadow has medium strength. The first frame, second frame, ground-truth for the second frame, and the final detection results can be seen in the first row, Figure (6a) to Figure (6d), with the object (only one object is considered) depicted in red and shadow depicted in green. The second sequence is the Pedestrian sequence, which is an outdoor scene with a human target, where the shadow has high strength. Similar results in the same order as the first sequence are shown in the figures of the second row. The third sequence is the Laboratory sequence, which is an indoor scene with a human target, and the shadow has low strength, where results are observed in the third row. The fourth sequence is the Saltshaker sequence, an indoor scene with a non-human target, where the shadow has medium strength, with results demonstrated in the fourth row. In the fifth sequence which is more complicated and is chosen from an animation, called the Wooden Model Sequence, the object casts more than one shadow in an indoor environment with medium to low strength. The shadows lie on both sides of the object in the. Finally, the Truck sequence in the seventh row shows two vehicles casting shadows on one side in an outdoor environment with medium to high strength. For this sequence and similar sequences where the foreground has more than one blob, binary foreground segmentation is needed and separate sliding windows need to be considered for each blob.

Although, evaluation of the method's efficiency is discussed in Section 3.2, a good match between ground-truth images in the third column with detection results in the fourth column can be seen.

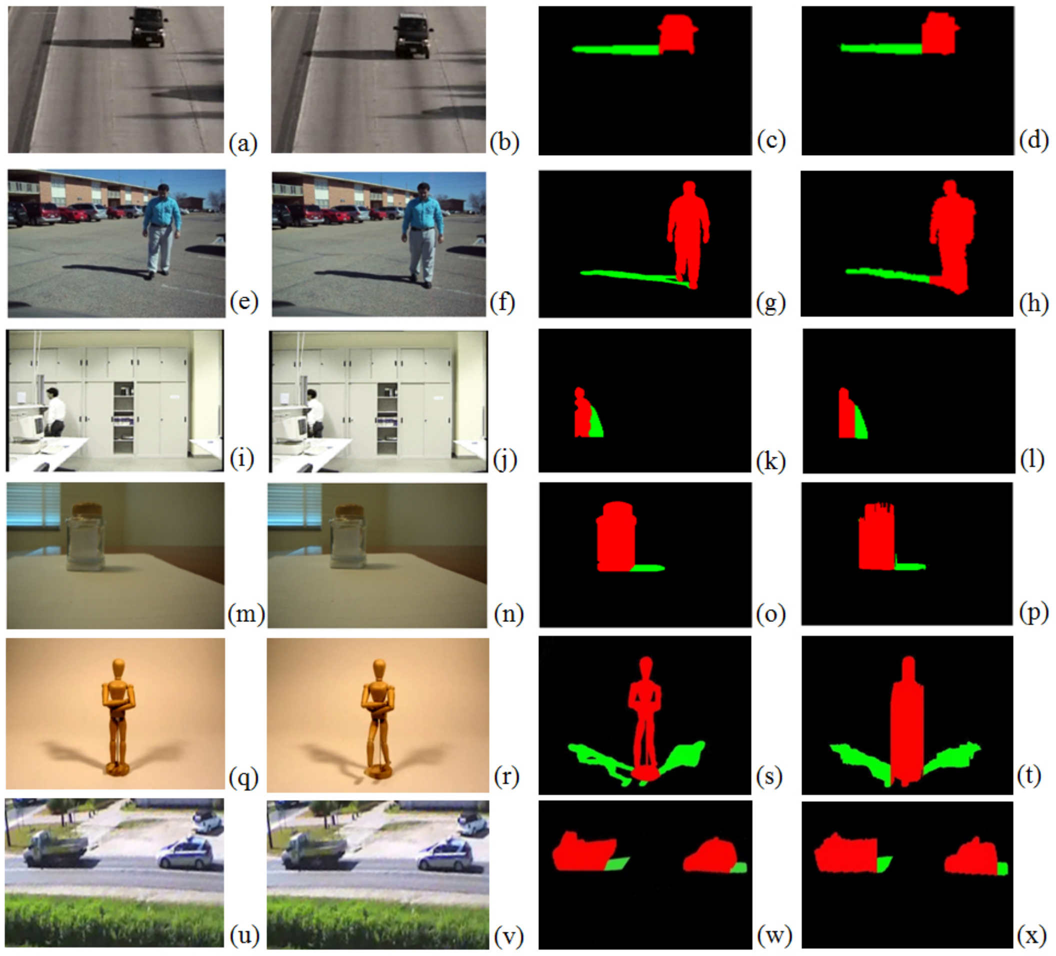


Figure 6. AHC qualitative results for moving cast detection. a) to d) First frame, second frame, ground-truth, and final shadow detection of the Highway sequence; e) to h) First frame, second frame, ground-truth, and final shadow detection of the Pedestrian sequence; i) to l) First frame, second frame, ground-truth, and final shadow detection of the Laboratory sequence; m) to p) First frame, second frame, ground-truth, and final shadow detection of the Saltshaker sequence; q) to t) First frame, second frame, ground-truth, and final shadow detection of the Wooden Model sequence; u) to x) First frame, second frame, ground-truth, and final shadow detection of the Truck sequence

To generate ground-truth images, first, the edge map of the second frame was delineated using the Canny edge detector with manually selected parameters determined to be the best. Next, extra edge pixels were manually eliminated, and non-connected edges were connected manually. Finally, image filling was performed on the accurate edge maps.

In complicated video sequences with occluding objects and shadows (see Figures 7a and 7b), multiple sliding windows can be used for one centre position of the window inside the foreground mask, as illustrated in image (c) of Figure 7. Each moving window belongs to a shadow-object combination, and a similar procedure as described above was performed for each individual window. As can be seen in image (d), final results correctly separate the foreground, and demonstrate the capability of the AHC method to handle shadow detection for multiple objects with occlusion.

We note that the method was shown to work for cases with non-smooth backgrounds.[28] We considered this case because we assumed in AHC that the normal vector to the background surface is fairly constant over the shadow region. If the background surface is mildly curved or is moderately non-smooth (such as a field of grass) the abrupt transition still exists and the method can perform well.

3.2 Quantitative results

To evaluate the accuracy of our proposed method, shadow detection rate η and shadow discrimination rate ξ proposed by Prati *et al.*[7] were used, given by (13) and (14), respectively:

$$\eta = \frac{TP_S}{TP_S + FN_S} \tag{14}$$

$$\xi = \frac{\overline{TP_F}}{TP_F + FN_F} \tag{15}$$

where TP indicates true positives, meaning the pixels detected correctly as belonging to a group, and FN indicates false negatives, meaning the pixels detected incorrectly as not belonging to a group. Subscripts "S" and "F" refer to shadow and foreground object, respectively. The term $\overline{TP_F}$ is the number of ground-truth pixels of the foreground objects minus the number of pixels detected as shadow, but belonging to foreground objects. Since these two metrics are used in most of the recent shadow detection methods and surveys,[8] we employ them here for comparison purposes.

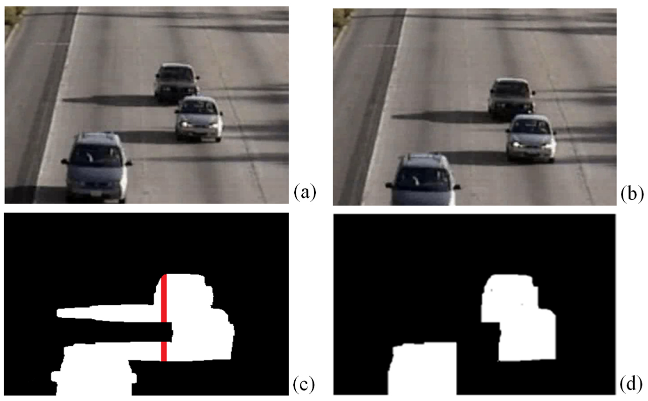


Figure 7. Handling occlusion scenarios in AHC. a) First frame of the Highway sequence with occlusion; b) Second frame of the Highway sequence with occlusion; c) Multiple sliding windows at a fixed centre position within the foreground mask; d) Final detection result

Table 1 summarizes image attributes for each sequence, including sequence type, shadow strength, and object class, in addition to efficiency metrics computed for seven sequences used in Figure 6. High individual and average metrics indicate that AHC works efficiently for different combinations of environment, target, and illumination condition. As can be seen, illumination and contrast do not affect the performance of AHC directly. For instance, shadow detection rate for a sequence with low contrast and low shadow strength (*e.g.* the Saltshaker Sequence) is higher than a sequence with high contrast and high shadow strength (*e.g.* the Truck Sequence). The reason is that in AHC, the difference between a shadow and an object region lies in the amount of correlation among luminance values, and because changing the illumination alters the luminance values of a region together, it does not change the correlation significantly.

As observed in Table 1, Pedestrian and Truck Sequences have the lowest η values, because a considerable number of shadow pixels were not detected as belonging to shadow (small $\overline{TP_S}$). This happened because in these sequences, due to using vertical scanning windows, shadow and foreground (and sometimes background) pixels were lying in the same window and correlation was calculated for a mixed region (shadow to the left of pedestrian and under his arm; shadow to the right of truck and under its bed). Shadow discrimination values on the other hand were not negatively affected because not so many foreground pixels were incorrectly detected as shadow (large $\overline{TP_F}$). Typically in AHC, when a window scans a mixed region of shadow and foreground object, the correlation values drop, the range measure increases, and the region is detected as a foreground region.

Table 1. Videos used, image attributes, and metrics for our AHC method

	Sequence Type	Shadow Strength	Object Class	η	ξ	
Highway Seq	Outdoor	Medium	Vehicle	95.9	98.8	
Pedestrian Seq	Outdoor	High	People	83.2	100	
Laboratory Seq	Indoor	Low	People	94.6	98.9	$\bar{\eta}$ = 85.4 $\bar{\xi}$ = 99.5
Saltshaker Seq	Indoor	Low	Object	93.4	99.4	
Wooden Model Seq	Indoor	Low	Object	81.5	99.8	
Truck Seq	Outdoor	High	Vehicle	63.7	100	

Table 2 compares the efficiency metrics from AHC method with the efficiency metrics from six other methods in the literature used in the recent survey of shadow detection methods,[8] all calculated for the Laboratory sequence. Among the competing methods in Table 2, Refs.[2, 10] are fast deterministic methods with medium accuracy, which employ background subtraction and the assumption of small change in hue when a shadow is cast on the background. As mentioned earlier, in many sequences the authors observed, this assumption is not valid, especially for low-quality surveillance videos. Furthermore, these methods use multiple empirically-determined thresholds for changes in hue, saturation, and value, which have to be experimentally determined based on the scene illumination conditions, and this is not feasible for recorded videos. The work[3] is more similar to the proposed AHC and provides relatively accurate results for benchmark videos; however it uses the assumption of colour constancy for a shadow region instead of luminance constancy, which is not always valid; it needs to have a fix background (not feasible) as well as multiple thresholds from data (higher computation time); and it employs segmentation for attaching multiple detached regions of shadow and object, while segmentation algorithms have intrinsic imperfection. The statistical method[9] for monochrome videos, while showing higher accuracy, is too complex, iterative, and requires training and accurate assumptions of many parameters. These limitations make this algorithm image dependent, environment dependent, and training dependent. It fails in the presence of multiple illumination sources and in outdoor environments with strong shadows. Authors in Ref.[12] utilized local, spatial, and temporal information for detecting shadows. While providing medium detection rates and high discrimination rates, this method has high computational complexity, requires information about the position of the sun with respect to the camera, and shows low illumination independence. The method by Stauder *et al.*[18] uses various heuristic techniques in order to exploit its main four assumptions. Results show an excellent detection and removal of indoor shadows. However, the limitations come from the fact that the approach is not applicable to outdoor shadows, and that it requires the background to be of a uniform colour. While the method in Ref.[29] provides relatively high accuracy measures for most of the scenes and conditions, it employs the assumption that the light energy received by a background point while not covered by a shadow is, to a high degree of approximation, affinely related to the energy it receives when a shadow is cast over it by an object. This assumption has not been validated by the authors and the real-time performance of the method has not been mentioned. Finally, the authors[30] used the change in chromaticity similar to Refs.,[2,10] but this statistical non-parametric method shows moderate higher accuracy at the expense of higher computational complexity.

Table 2. Results of different shadow detection methods in Ref.[8] and our AHC method applied to the Laboratory sequence

Method	η	ξ
Horprasert *et al.* [10]	84	92.4
Stauder *et al.* [18]	60.3	81.6
Mikic *et al.* [12]	64.9	95.4
Cucchiara *et al.* [2]	76.3	89.9
Al-Najdawi *et al.* [29]	90.2	92.8
Jung [9]	85.8	95.1
AHC	**94.6**	**98.9**

As can be seen, our AHC method shows improved performance compared with other methods in terms of both detection and discrimination rates. We believe it notable that AHC can be employed in real-time situations without limiting assumptions, manual selection of critical parameters, need for prior knowledge, or modelling complexity. No other competing method in this table has demonstrated real-time performance. A fair comparison in accuracy is only possible when computational times are comparable. Also as mentioned earlier, some methods require further external information to provide high accuracy and have to be used in specific conditions and applications.

Table 3 provides another comparison between the efficiency metrics from AHC with the efficiency metrics from five other methods in the literature, calculated for the Highway sequence. The first method is the work by Amato *et al.*[3] mentioned in Section 2.2, and four others are presented in the survey of shadow detection methods by Prati *et al.*[7] As can be seen, our AHC method outperforms other methods.

Table 3. Results of different shadow detection methods in Refs.[3,7] and AHC method applied to the Highway sequence

Method	η	ξ
Amato *et al.* [3]	81.0	85.4
Haritaoglu *et al.* [31]	81.6	63.8
Mikic *et al.* [12]	59.6	84.7
Cucchiara *et al.* [2]	69.7	76.9
Stauder *et al.* [18]	79.5	62.4
AHC	**95.9**	**98.8**

While AHC outperforms the other methods in Tables 2 and 3, comparing the results reveals that for the Highway Sequence, this performance difference is more considerable as this sequence has a very low quality and negligible colour information upon which some of the methods relied (*e.g.* Ref.[2]). Methods designed to work with monochromatic information,[9] exhibit better performance for these benchmark sequences.

Note that due to unavailability of codes for other algorithms, we were not able to apply them to all sequences used in this research and could not provide statistical measures for η and ξ, such as average or standard deviation for each algorithm.

3.3 Discussion of the Results

One issue is selection of the width for the sliding window (N). Although $N = 8$ pixels was used for all figures in Section 3.1 and all tables in Section 3.2, here, we try to elaborate more on how we selected $N = 8$ pixels. As we expect, increasing N would reduce the accuracy of transition point localization. This accuracy reduction is not significant as long as the width of the window is included in calculation of the transition point location. Also as we expect, increasing N leads to small increment of the computation time (here using MATLAB) since more window locations has to be scanned. Figure 8 shows the effect of change in the window's width, N, on the inverse of computation time (frames per second, fps) and shadow detection accuracy for the Pedestrian Sequence (here, only η is considered since ξ remains 100% for this sequence). The value $N = 8$ selected for HAC was found to be a trade-off between accuracy and computation time.

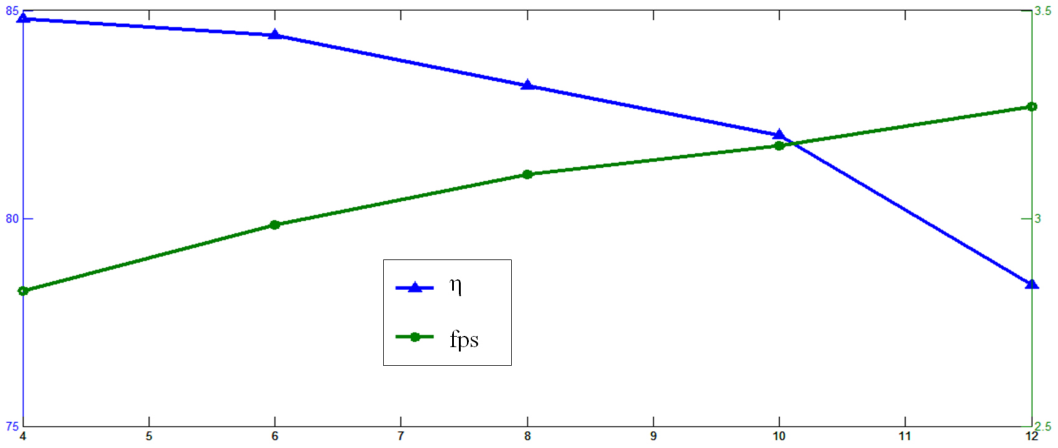


Figure 8. Effect of width for the sliding window on shadow detection rate and inverse of the computation time (frames per second)

Selecting the amount of shift used for the sliding window location (d) to calculate $Corr_x$ values in (8) is another topic that has to be discussed. When d becomes larger, the correlation between the luminance values within adjacent windows would decrease regardless of the location of the sliding window. Therefore the transition of Corrx values between the shadow and the object region fades. In parts (a) to (c) of Figure 9, $Corr_x$ values for the Pedestrian Sequence filtered by a moving average filter (filter is used to smooth the oscillations and better render the transition in the plot) denoted by $Corr_x^f$ are shown versus the location of the sliding window for $d = 1$, $d = 3$, and $d = 5$ pixels, respectively. As can be seen, when d increases, correlation values for windows within the shadow region start to decrease and so, the transition become less abrupt and vivid. Therefore, we opted to use $d = 1$ pixel for AHC.

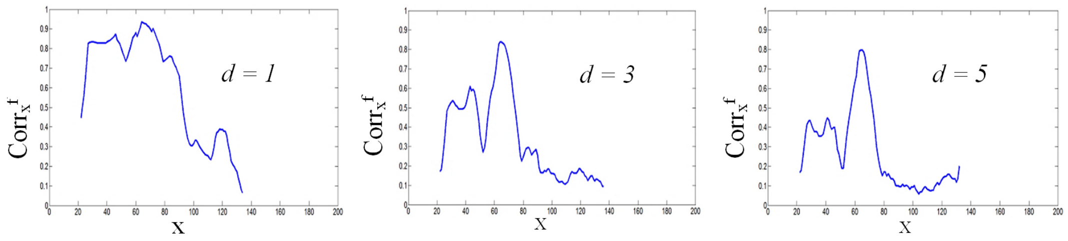


Figure 9. Effect of shift for the sliding window, d in (8) on filtered correlation values. a) $d = 1$ pixel shift was used; b) $d = 3$ pixels shift was used; c) $d = 5$ pixels shift was used

The final discussion in this section is about the orientation of the sliding window. When the cast shadow does not lie to the right or left of the object, scanning using vertical sliding windows as we did so far will not detect the shadow region as we discussed using part (c) of Figure 4. The Surveillance Camera Sequence in Figure (10a) and (10b) demonstrates such situation. As shown in part (c) of this figure, vertical sliding windows will only detect the portion of shadow that is within the red rectangle. Shadow pixels within the green rectangle will be inside one window with object pixels and due to small modified correlation values for these window locations, they will be misclassified as the object pixels. This is while horizontal scanning windows can be more helpful in shadow detection in such sequence where the shadow lies to the top or bottom of the object. This is demonstrated in part (d), where vertical sliding windows can only detect the shadow pixels in the blue and the hatched regions, while horizontal windows can detect significantly more shadow pixels (white region+hatched region). The suggestion from this discussion is to use both vertical and horizontal sliding windows in AHC for any sequence. In horizontal windows, the height is fixed and width will be adaptable to the left and right boundaries of foreground mask. Here, if both horizontal and vertical windows are used for scanning, η will increase from 26% to 79%, while ξ will stay 100%. This is because a large portion of shadow lies to the bottom of the vehicle and it can only be detected using horizontal scanning windows.

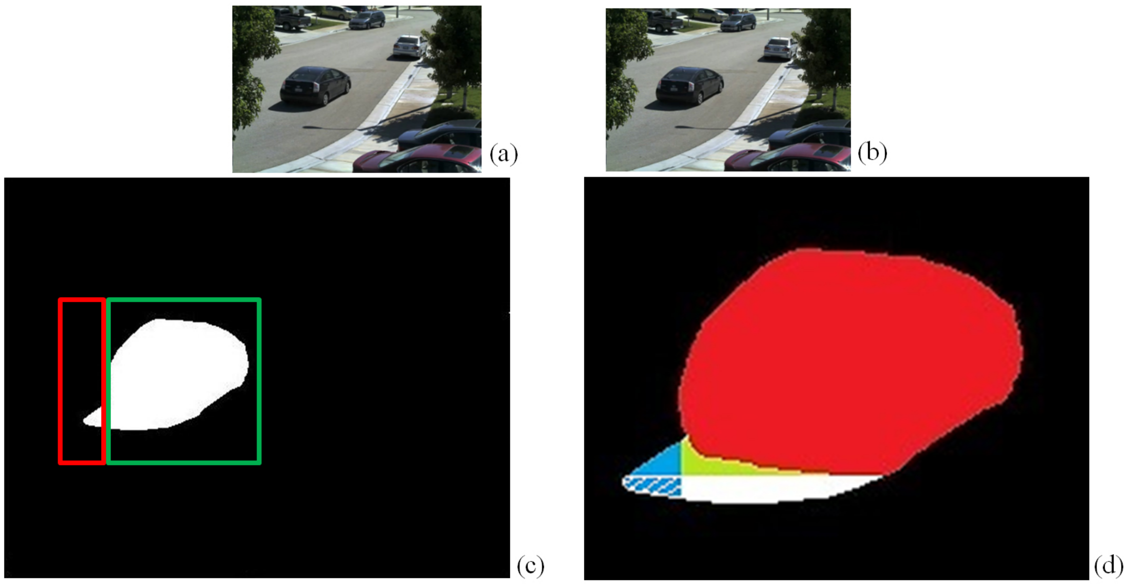


Figure 10. Using vertical and horizontal sliding windows for the Surveillance Camera Sequence. a) First frame; b) Second fame; c) Foreground mask: shadow pixels within the red rectangle will only be correctly detected as shadow by vertical sliding windows; shadow pixels within the green rectangle will be misclassified as object pixels by vertical sliding windows; d) Blue region: shadow pixels detected by vertical sliding windows only; White region: shadow pixels detected by horizontal windows only; Hatched region: shadow pixels detected by vertical and horizontal windows; Green region: shadow pixels not detected by vertical or horizontal windows

Note that shadow pixels within the green region will not be detected by either type of window mentioned above. Further processing, such as searching for pixels in the object region which have similar hue and luminance values in a neighbourhood of the transition point with a given radius might detect the remainder of such shadow pixels at the expense of computation time increase as mentioned in Ref.[28] Devising a searching algorithm which can detect a significant number of remaining shadow pixels, but still allows real-time application of AHC is left as a future work.

4. Real-time implementation

To analyse the performance of the AHC method, we implemented the algorithm in C++ using the OpenCV library. We then examined the algorithm's run-time requirements on a modern desktop computer for the six sequences in Figure 6. The details of apparatus for this investigation are provided in Table 4.

4.1 Software implementation and test images

The AHC method was implemented in C++ by using a combination of the C++ Standard Library and OpenCV. All source and intermediate image data were represented in memory as OpenCV Mat objects. Other one-dimensional data arrays were represented in memory by using the C++ vector class.

Creation of the foreground mask (Section 2.1) was performed entirely in OpenCV via direct pixel access (for the thresholding); via a combination of the findContours() and drawContours() functions for the image filling; via the morphologyEx() function for the image dilation; and via the GaussianBlur() and Canny() functions for the edge detection. Detection of the shadow (Section 2.2) was performed via custom C++ code which used methods and associated functions of the vector class and direct pixel access to the Mat objects.

Table 4. Hardware and software specifications for the performance analysis

Specification	Value
C++ Environment	Visual Studio Professional 2013 Version 12.0.21005.1
OpenCV Version	2.4.10
OS Version	Windows 7 Enterprise 64-bit SP1
Processor Model	Intel Core i7-4470
Processor Frequency	3.4 GHz
Processor Cores	8 Cores
System RAM	8 GB Dual-Channel DDR3 @ 800 MHz

It is important to note that no explicit attempts were made to parallelize the implementation to take advantage of the multiple cores available on the test system. Accordingly, during the analysis, we observed that only one of the cores (sometimes, but rarely, two of the cores) was active at any given time.

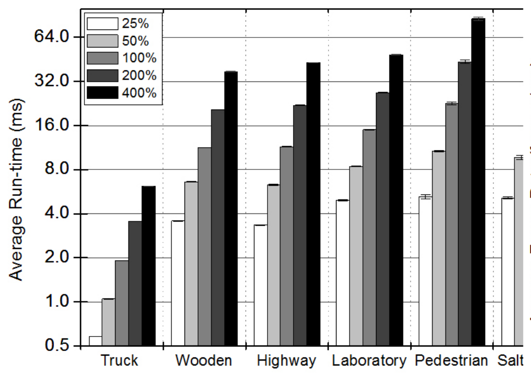
Table 5. Native and resized dimensions of the six sequences used in the performance analysis; the percentages denote total numbers of pixels

Sequence	25%	50%	100% (native)	200%	400%
Truck	146×70	207×99	292×140	413×198	584×280
Wooden	240×180	340×255	480×360	679×510	960×720
Highway	320×203	453×287	640×405	906×573	1280×810
Laboratory	320×240	453×340	640×480	906×679	1280×960
Pedestrian	320×240	453×340	640×480	906×679	1280×960
Saltshaker	320×240	453×340	640×480	906×679	1280×960

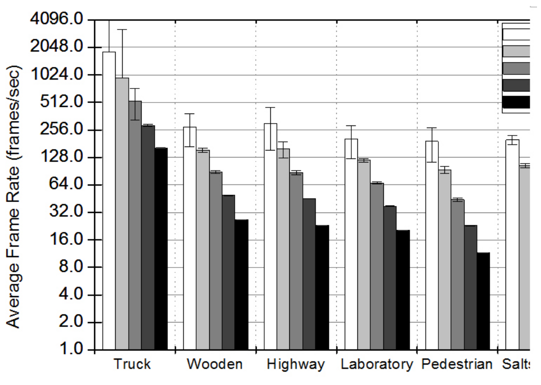
Six sequential-image-pair sequences were used for the performance analysis: Highway, Pedestrian, Laboratory, Saltshaker, Truck, and Wooden. All of the sequences originally contained 24-bit colour images spanning a range of spatial sizes and shadow intensities and extents. To evaluate the effects of image size on the run-time, each sequence was resized by factors of 50%, 70.7%, 141.4%, and 200% in each dimension, corresponding to 25%, 50%, 200%, and 400% of the total number of native pixels, respectively. The native and resized dimensions of each sequence are listed in Table 5.

4.2 Timing results

For each sequence, at each size, we ran the AHC implementation 100 times. The means and standard deviations of the 100 trials (in milliseconds) are shown in Figure 11 for each sequence at each size. Note that the sequences were resized and converted to grayscale prior to the timing; thus, the data shown in Figure 11 do not include these pre-processing times, nor do they include the times required to load the images from disk.



(a)



(b)

Figure 11. Computation time of AHC on test sequences of various sizes. The value of each bar denotes the average over 100 trials; error bars denote standard deviations over these 100 trials. Note that the vertical axis is on a logarithmic scale. a) Average run-time in ms; b) Average frame rate in frames/sec

Observe from Figure (11a) that for all of source sequences, the average run-time for AHC increases linearly with the number of pixels. Specifically, for any given source sequence, doubling the number of pixels approximately doubles the run-time. Furthermore, the source sequences are ordered from left to right in Figure (11a) in terms of an increasing number of pixels (Trucks has the least number of pixels; Laboratory, Pedestrian, and Saltshaker have the most number of pixels), and the corresponding run-times also generally increase in a left-to-right fashion for any given scaling factor (any given coloured bar). For example, at a 50% scaling factor, Laboratory, Pedestrian, and Saltshaker contain 307,200 pixels, which is 7.5 times the number of pixels of Trucks at a 50% scaling factor (40,880 pixels). Accordingly, the former three sequences require approximately 8x, 10x, and 9x the run-time of the latter.

However, it is also evident from Figure (11a) that the run-time is not independent of the source sequence. In particular, observe that the last three sequences (Laboratory, Pedestrian, and Saltshaker) exhibit a variation in run-time for any given scaling factor, despite the fact that these sequences have equal sizes. Nonetheless, as shown in Figure (11b), even at a size of 1,280 × 960 pixels, the implementation can achieve a processing rate in excess of 10 frames/sec. An increase in the processing rate could possibly be achieved via strategic parallelization (*e.g.*, processing separate frames on different cores).

Figure 12 shows the distributions of the run-times required for the various stages for each sequence at its native size (100% scaling; see Table 5). For each pie chart, the slice shown in green denotes the run-time required for the segmentation (Section 2.1), and the slice shown in light blue denotes the run-time required for the correlation, range, and other statistical computations of the shadow detection (Section 2.2). Roughly, these data suggest that the run-times required for the segmentation *vs.* shadow detection depend on the size of the shadow; for Truck and Laboratory, the shadow is small, and thus the segmentation consumes the majority of the run-time. For the other sequences, the segmentation vs. shadow-detection run-times are nearly equal. In regards to the segmentation, the majority of the run-time is attributable to either the image filling or the Canny edge detector.

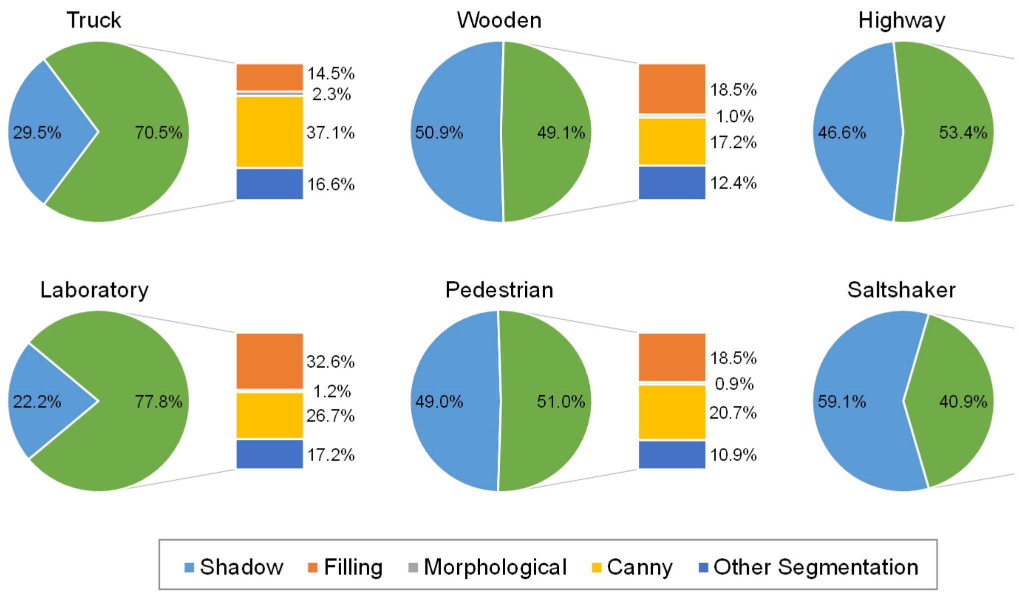


Figure 12. Average run-time of each stage of AHC on the test sequences at their natives sizes (100%)

In summary, the proposed AHC method can achieve real-time performance, even for relatively large sequences, when run on a modern desktop computer. Furthermore, for the sequences tested here, at least 50% of the run-time is attributable to the segmentation stage; and, as we mentioned in Section 2, this stage could be greatly simplified given a fixed background setting. A further reduction in run-time could possibly be achieved by taking advantages of multiple cores. The most straightforward approach would be to process sequential frames on separate cores, assuming that the frame rate of the capture device is faster than the processing time. In regards to per-frame parallelization, although the segmentation stage is not easily parallelized due to the dependency between its sub-stages, and due to the spatial dependency of its operations (*e.g.*, filtering during Canny edge detection), the statistical computations of the shadow-detection stage are quite straightforward to distribute across multiple cores.

5. Limitations and future work

A fast foreground detection method, which reduces the dependence of the method's efficiency on the edge map results, is preferred over the combination of image differencing and edge detection. If the shadow region is separated from the object region by a concave boundary, AHC will misidentify some shadow pixels, and efficiency will be reduced at the transition point. Devising a searching algorithm which can detect a significant number of remaining shadow pixels, but still allows real-time application of AHC is left as a future work.

6. Conclusion

In this research, a new real-time approach for detection of moving cast shadows (called the AHC method) is presented. While avoiding the need for a-priori knowledge, limiting structures, unrealistic assumptions, and non-robust features, the AHC method demonstrates highly accurate detections and provides improved performance compared with other shadow detection methods in the literature. A deterministic nonmodel-based approach, employing 2D luminance scanning provides a fast detection, which is useful for real-time situations. Finally, an unsupervised real-time implementation in C++ using OpenCV library shows possibility for application of the method to real-time tasks, such as surveillance and tracking systems, remote sensing, and border security.

REFERENCES

[1] Arbel E, Hel-Or H. Shadow Removal Using Intensity Surfaces and Texture Anchor Points. IEEE Trans. Pattern Anal. 2011 June; 33(6).

[2] Cucchiara R, Grana C, Piccardi M, *et al*. Detecting Moving Objects, Ghosts, and Shadows in Video Streams. IEEE Trans. Pattern Anal. 2003; 25(10): 1137-342.

[3] Amato A, Mozerov MG, Bagdanov AD, *et al*. Accurate Moving Cast Shadow Suppression Based on Local Colour Constancy Detection. IEEE Trans. Image Process. 2011; 20(10): 2954-66.

[4] Subramanyam M, Nallaperumal K, Ravi S. A Review on Survey and Analysis of Shadow Detection Techniques. 2014.

[5] Sanin A, Sanderson C, Lovell BC. Shadow detection: A survey and comparative evaluation of recent methods. Pattern Recognition. 2012; 45(4): 1684-95. http://dx.doi.org/10.1016/j.patcog.2011.10.001

[6] Hu JS, Su TM, Jeng SC. Robust Background Subtraction with Shadow and Highlight Removal for Indoor Surveillance. In: Proc. 2006 IEEE/RSJ Int. Conf. Intelligent Robots and Systems; October 2006. p. 4545-50.

[7] Prati A, Mikic I, Trivedi M, *et al*. Detecting Moving Shadows: Algorithms and Evaluation. IEEE Trans. Pattern Anal. 2003; 25(7): 918-23.

[8] Al-Najdawi N, Bez HE, Singhai J, *et al*. A survey of cast shadow detection algorithms. Pattern Recog Lett. 2012; 33(6): 752-64. http://dx.doi.org/10.1016/j.patrec.2011.12.013

[9] Jung CR. Efficient Background Subtraction and Shadow Removal for Monochromatic Video Sequences. IEEE Trans. Multimedia. 2009; 11(3): 571-7. http://dx.doi.org/10.1109/TMM.2009.2012924

[10] Horprasert T, Harwood D, Davis LS. A Statistical Approach for Real-Time Robust Background Subtraction and Shadow Detection. Proc. IEEE I. Conf. Comp Vis, FRAME-RATE Workshop; 1999.

[11] Trivedi MM, Mikic I, Kogut G. Distributed Video Networks for Incident Detection and Management. Proc. IEEE Int. Conf. Intell Transp; 2000 Oct. p. 155-60.

[12] Mikic I, Cosman P, Kogut G, *et al*. Moving Shadow and Object Detection in Traffic Scenes. IEEE Int. Conf. CVPR; 2000. p. 321-4.

[13] Bi S, Liang D, Shen X, *et al*. Human Cast Shadow Elimination Method Based on Orientation Information Measures. Proc. IEEE Int. Conf. Automation and Logistics; 2007. p. 1567-71.

[14] Chung KL, Lin YR, Huang YH. Efficient Shadow Detection of Colour Aerial Images Based on Successive Thresholding Scheme. IEEE Trans. Geosci Remote S. 2009; 47(2): 671-82. http://dx.doi.org/10.1109/TGRS.2008.2004629

[15] Hsieh J, Hu W, Chang C, *et al*. Shadow Elimination for Effective Moving Object Detection by Gaussian Shadow Modelling. Int. J. Image Vision Comput. 2003; 21: 505-16.

[16] Onoguchi K. Shadow Elimination Method for Moving Object Detection. Int. Conf. Patt Recog. 1998; 1: 583-7.

[17] Russell M, Zou JJ, Fang G. Real-time vehicle shadow detection. Electronics Letters. 2015; 51(16): 1253-5. http://dx.doi.org/10.1049/el.2015.1841

[18] Stauder J, Mech R, Osterman J. Detection of Moving Cast Shadows for Object Segmentation. IEEE Trans. Multimedia. 1999; 1(1): 65-76. http://dx.doi.org/10.1109/6046.748172

[19] Nadimi S, Bhanu B. Physical Models for Moving Shadow and Object Detection in Video. IEEE Trans. Pattern Anal. 2004; 26(8): 1079-87. PMid:15641737. http://dx.doi.org/10.1109/TPAMI.2004.51

[20] Amato A, Mozerov MG, Casado IH, *et al*. Background Subtraction Technique Based on Chromaticity and Intensity Patterns. In: Proc. 19th IEEE Int. Conf. Patt Recog, Tampa, FL; Dec. 8-11, 2008. p. 1-4.

[21] Kar A, Deb K. Moving cast shadow detection and removal from Video based on HSV color space. Electrical Engineering and Information Communication Technology (ICEEICT), 2015 International Conference on. IEEE; 2015.

[22] Huerta I, *et al*. Chromatic shadow detection and tracking for moving foreground segmentation. Image and Vision Computing. 2015; 41: 42-53. http://dx.doi.org/10.1016/j.imavis.2015.06.003

[23] Sangwine SJ, Robin ENH, eds. The colour image processing handbook. Springer Science & Business Media; 2012.

[24] Poynton C. Digital video and HD: Algorithms and Interfaces. Elsevier; 2012.

[25] Watt A. 3D Computer Graphics. Reading, MA: Addison-Wesley; 1993.

[26] Warren JS. Modern Optical Engineering. McGraw-Hill; 2007. p. 228.

[27] Joshi A, Papanikolopoulos N. Learning to Detect Moving Shadows in Dynamic Environments. IEEE Trans. Pattern Anal. 2008; 30(11): 2055-63. PMid:18787252. http://dx.doi.org/10.1109/TPAMI.2008.150

[28] Andalibi M. Robust Real-Time Detection of Moving Cast Shadows. Ph.D. Qualifying Exam Report, School of Mechanical and Aerospace Engineering, Oklahoma State University, Stillwater, Oklahoma, U.S.A.; 2013. p. 33-5.

[29] Al-Najdawi N, Bez H, and Edirisinghe E. A novel hypothesis for cast shadow modelling and detection for objects tracking and recognition. Proc. 6th IASTED Conf. on Visual Imag and Image Process; 2006. p. 377-82.

[30] Haritaoglu I, Harwood D, Davis LS. W4: Real-Time Surveillance of People and Their Activities. IEEE Trans. Pattern Anal. 2000; 22(8): 809-30. http://dx.doi.org/10.1109/34.868683